STEINER NETWORK CONSTRUCTION FOR SIGNAL NET ROUTING WITH DOUBLE-SIDED TIMING CONSTRAINTS

A Thesis

by

QIUYANG LI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2006

Major Subject: Computer Engineering

STEINER NETWORK CONSTRUCTION FOR SIGNAL NET ROUTING WITH DOUBLE-SIDED TIMING CONSTRAINTS

A Thesis

by

QIUYANG LI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Jiang Hu
Committee Members,	Gwan Choi
	Donald K. Friesen
Head of Department,	Costas N. Georghiades

August 2006

Major Subject: Computer Engineering

ABSTRACT

Steiner Network Construction for Signal Net Routing with

Double-sided Timing Constraints.

(August 2006)

Qiuyang Li, B.S., Nankai University;

M.S., Nankai University

Chair of Advisory Committee: Dr. Jiang Hu

Compared to conventional Steiner tree signal net routing, non-tree topology is often superior in many aspects including timing performance, tolerance to open faults and variations. In nano-scale VLSI designs, interconnect delay is a performance bottleneck and variation effects are increasingly problematic. Therefore the advantages of non-tree topology are particularly appealing for timing critical net routings in nano-scale VLSI designs. We propose Steiner network construction heuristics which can generate either tree or non-tree of signal net with different slack wirelength tradeoffs, and handle both long path and short path constraints. Extensive experiments in different scenarios show that our heuristics usually improve timing slack by hundreds of pico seconds compared to traditional tree approaches while increasing only slightly in wirelength. These results show that our algorithm is a very promising approach for timing critical net routings.

ACKNOWLEDGEMENTS

This thesis represents about one and a half years of work at Texas A&M University. This work would not have been possible without the boundless assistance of mentors, colleagues, and friends.

It is with the deepest of gratitude that I would like to thank my advisor, Jiang Hu, who always gave me great directions. I would also like to express my heartfelt thanks to the other professors, who have always kept their doors open, ready to discuss and encourage new ideas. Particular thanks go to Gwan Choi and Donald K. Friesen who could always find time, despite any other responsibilities or deadlines they may have had.

In my time at Texas A&M University, I have had the opportunity to collaborate with a number of different people without whose help this thesis could not have been completed. I would like to thank Weiping Shi, Sunil P. Khatri, Di Wu, Wei Zhuang, Mankang Mai, and Chin-Ngai Sze (Cliff) for their time and their efforts. While not a collaborator as such, I would also like to thank Tammy Carda and Linda Currin for their regular guidance and tireless assistance.

Finally, I would like to express my sincere thanks to friends and family whose unwavering support has been indispensable to me over the last years. Thank you to my family: Dad, Mom, my elder brother and sister. And thank you to the friends that I have made along the way: Mankang, Wei Zhuang, Dawen, Xiaomin, Ying Li, Jiong Yan and Shengquan.

TABLE OF CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

The interconnect delay is a well-known performance bottleneck in VLSI circuit designs. Therefore for timing optimization, the optimization of interconnect topology, say singal net routing, is very important for the circuit design.

In practice, because Steiner tree [1] is cost-effective and its delay is relatively easy to compute, people almost always use it for signal net routing. However, non-tree topology has some remarkable advantages compared to trees. Non-tree routing can significantly improve signal propagation delay, reduce signal skew, and afford increased reliability with respect to open faults that may be caused by manufacturing defects and electro-migration [3]. That is, the redundant paths in a non-tree network provide certain tolerance to open faults and therefore can improve manufacturing yield and reliability [2]. Moreover, non-tree topology sometimes can reduce delay variations [2]. Although non-tree delay computation is more expensive than that of trees, the computation overhead of non-tree can usually be alleviated by the advancement on computation techniques and facilities. And the design needs often eventually outweigh computation overhead if the overhead is not prohibitively large.

1.1 Previous Work

Perhaps the first non-tree routing work is [3]. It starts with a Steiner tree topology. Then it iteratively searches for a new edge to add, so that the maximum source-sink delay in the resulting routing graph will be minimized. It keeps on doing this until no further delay improvement is possible. The later work of [4] inserts links sequentially between the

---

This thesis follows the style of *IEEE Transactions on Microwave Theory and Techniques*.

source and the sink with the maximum delay in the topology with shortest feasible length. The recent work of [2] is focused on the reliability and manufacturing yield of non-tree routing. It augments extra edges to an existing tree to increase the percentage of 2-connected wires, which implies tolerance to open faults. The works of [3, 4] on timing driven non-tree routing have two main weaknesses. Since they start from an existing tree, and then add wires on it, the performance of the resulting non-trees depends on the initial trees. The arbitrary starting tree cannot guarantee a good non-tree solution. The other weakness is that they [3, 4] optimize only delay without considering timing constraints. In reality, maximizing slack or minimizing wire cost subject to timing constraints is a more common and useful problem formulation [5].

The timing constraints in previous works [5] almost always consider only the upper bounds for sink delays. In fact, there are delay lower bounds due to the short path (hold time) constraints in synchronous circuits. Some gate sizing works [6] consider both delay upper bound and lower bound at the same time. To the best of our knowledge, there is no signal net routing work considering the double-sided timing constraints yet. This is perhaps due to the reason that delay lower bound can be easily satisfied by padding extra delay. The delay padding can be implemented by wire detour, adding dummy capacitors or inserting redundant buffers. The former two approaches may increase the delay along the long path. The later approach of redundant buffers may intensify the leakage power problem. They all can increase the unnecessary complexity. Thus, we need to handle the short path constraints in a more careful manner.

1.2 Outline

In this thesis, we propose Steiner network construction heuristics which consider delay upper bound and lower bound simultaneously for timing critical nets. We will show that sometimes a link insertion can simultaneously reduce long path delay and increase short path delay. One heuristic is a greedy link insertion in an existing tree or non-tree, which is similar to [3] but the solution search is trimmed for the double-sided timing constraints. The other is a dynamic programming based constructive algorithm which can generate a set of solutions with different slack-wirelength tradeoff and can reach either tree or non-tree topology.

By comparing to the traditional AHHK tree results, our extensive experimental results show that this Steiner network construction usually improves slack by hundreds of pico seconds. The non-tree approach may bring some wirelength and runtime overhead, but from the experimental results, this overhead is in a relatively small range. And moreover, because it is applied to only a small number of timing critical nets, the impact of overhead to overall chip design is very limited. Beside this, we also do the Monte Carlo simulation with process variations considered. The results show that our method can improve timing yield greatly with both nominal slack improvement and delay variability (standard deviation) reduction.

The rest of this thesis is organized as following. Chapter II discusses a lemma of link insertion and then gives our problem formulation. Chapter III addresses our algorithm. In Chapter IV, we show our experiment results. And then we conclude in Chapter V.

CHAPTER II

PRELIMINARY

In this chapter, we will show that proper link insertion in an existing tree or nontree can reduce long path delay and increase short path delay simultaneously. That is, link insertion may reduce the difference of the maximum path delay and the minimum path delay.

Considering insert a link between two nodes i and j in an RC network (Fig.1), which can be either a tree or a nontree. Let the link resistance be R and link capacitance be C. According to the $\prod$-model, this link insertion is equivalent to adding capacitance $C/2$ at node i and j, respectively, and inserting resistance R between i and j.

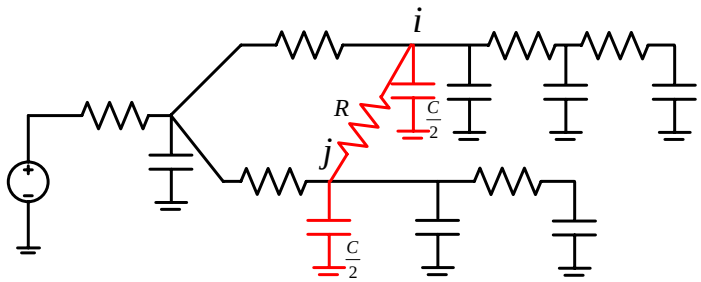


Fig.1 Insert a link in an RC network.

The link capacitance always increases the delay by $t_{i,lc}=\frac{C}{2}(R_{i,i}+R_{i,j})$ and $t_{j,lc}=\frac{C}{2}(R_{i,j}+R_{j,j})$. Here $R_{i,i}(R_{j,j})$ is the path resistance from the source to node $i(j)$. And $R_{i,j}$ is the transfer resistance which equals the voltage at node i when 1A current is

injected into node j and all the other node capacitances are set to zero [4]. After the link insertion, the delay to i and j are changed from t_i and t_j to $\tilde{t}_i$ and $\tilde{t}_j$ according to the following equations [7]:

$$\tilde{t}_i = (1-\alpha)(t_i + t_{i,lc}) + \alpha(t_j + t_{j,lc}) \qquad (1)$$

$$\tilde{t}_j = (1-\beta)(t_j + t_{j,lc}) + \beta(t_i + t_{i,lc}) \qquad (2)$$

Where $\alpha = \frac{r_i}{R + r_i - r_j}$ and $\beta = \frac{r_j}{R + r_i - r_j}$. In general, r_i and r_j are equal to the Elmore delay at i and j, respectively, when node capacitance $C_i = 1, C_j = -1$ and the other node capacitances are set to zero [4].

The above equations show that the link capacitance always increases signal delay while the link resistance attempts to average the delay between i and j. It is strightforward to derive the following condition on the simultaneous improvement for both long path and short path delay.

Lemma: If a link with resistance R and capacitance C is inserted between a node i on a long path and a node j on a short path in a Steiner network, the necessary and sufficient condition of simultaneously reducing delay to node i and increasing delay to node j is $t_i \geq (\frac{1}{\alpha} - 1)t_{i,lc} + t_j + t_{j,lc}$.

When considering double sided timing constraints, each sink v_i has a delay upper bound $\overline{q}_i$ and a delay lower bound $\underline{q}_i$. The delay upper bound is the same as the required arrival time (RAT) in traditional methods. We define the **late slack** of a sink v_i as

$\overline{s}_i = \overline{q}_i - t_i$ where t_i is the delay. Similarly, the **early slack** of a sink v_i is defined as $\underline{s}_i = t_i - \underline{q}_i$. The **slack** of a sink v_i is $s_i = \min(\overline{s}_i, \underline{s}_i)$. The late slack, early slack and slack of a network (or subnetwork) are the minimum late slack, early slack and slack among all sinks in the network, respectively. For a network (or subnetwork), the sink having the minimum late (early) slack is called **late (early) critical sink**. Here is our problem formulation:

Timing Driven Steiner Network Construction:

Given a source node v_0, a set of sink nodes $\{v_1, v_2, ..., v_n\}$ with each sink v_i having load capacitance c_i, lower delay bound $\underline{q}_i$ and upper delay bound $\overline{q}_i$, construct a rectlinear Steiner network spanning the source and the sinks such that the slack of the network is maximized.

CHAPTER III

ALGORITHM

Before we dive into the details of the algorithm, let's review our problem formulation.

Given a source node v_0, a set of sink nodes $\{v_1, v_2, ..., v_n\}$ with each sink v_i having load capacitance c_i, lower delay bound $\underline{q}_i$ and upper delay bound $\overline{q}_i$, construct a rectlinear Steiner network spanning the source and the sinks such that the slack of the network is maximized.

Then comes the procedure of our algorithm, constructive Steiner network heuristic.

Step 1. *Initialization*: A set of subnetworks are initialized with the sink nodes. It is the first candidate solution.

n = number of sinks
O_0 = new empty solution
for $i = 1$ to n
    G_i = new empty subnetwork
    add sink node v_i to G_i
    add G_i to O_0
add O_0 to solution set O

Step 2. *Merging selection*: In a candidate solution O_i, select two subnetworks to merge.

For two different scenarios: (1) If long path constraints and short path constraints are almost equally tight, we first choose the subnetwork with the maximum $(\overline{q} + \underline{q})/2 + t'$,

and then merge it with its nearest neighboring subnetwork. (2) If long path constraints dominate, we choose a pair of subnetworks whose merging root is farthest from the source among all pairs.

for every subnetwork G in O_k

Choose G_i with maximum $(\overline{q} + \underline{q})/2 + t'$

for every other subnetwork G in O_k

Choose G_j with minimum $|x_j - x_i| + |y_j - y_i|$

call Step 3 to merge G_i and G_j

n_k = the number of subnetworks in O_k

max =*i* = *j* = 0

for i = 1 to n_k

for j = i to n_k

if $|x_j - x_i| + |y_j - y_i|$ > *max then* *m=i; n = j*

call Step 3 to merge G_m and G_n

Step 3. *Merging*: Merge these two subnetworks.

Use two different method to merge two selected subnetwoks G_i and G_j of solution O_k. One is root-root merging. And the other method is shortest merging where two nodes from the two subnetworks with the minimum distance are connected directly. By doing this, we get two candidate solutions.

O_m = new empty solution

O_m = O_k

G_m = new empty subnetwork

node set of G_m = { node set of G_i } $\cup$ { node set of G_j }

edge set of G_m = { edge set of G_i } $\cup$ { edge set of G_j }

v_m = new node with coordinate { $median(x_i, x_j, x_o)$, $median(y_i, y_j, y_o)$ }

add v_m to G_m and it is the new root

add edges $\{v_i, v_m\}$ and $\{v_j, v_m\}$ to G_m

delete G_i and G_j in O_m

call Step 4 to insert link in G_m

O_n = new empty solution

$O_n = O_k$

G_n = new empty subnetwork in O_n

node set of G_n = { node set of G_i } $\cup$ { node set of G_j }

edge set of G_n = { edge set of G_i } $\cup$ { edge set of G_j }

min = ∞

for every node v_i in G_i

for every node v_j in G_j

if $|x_j - x_i| + |y_j - y_i| < min$ *then* $p=i; q = j$

v_n = new node with coordinate { $median(x_p, x_q, x_o)$, $median(y_p, y_q, y_o)$ }

add v_n to G_n and it is the new root

add edges $\{v_i, v_n\}$ and $\{v_j, v_n\}$ to G_n

delete G_i and G_j in O_n

call Step 4 to insert link in G_n

delete old solution O_k

Step 4. *Link insertion*: Insert link into the result subnetwork G of solution O_k .

For the two subnetworks obtained from mergings, we insert a link in each of them. Then we get two new candidate solutions.

$O_m = O_k$ // the following operations are taken in this new solution O_m

v_e is the early critical sink of G

v_l is the late critical sink of G

use dijkstra's algorithm to find the shortest path $p_{e,l} \in G$ which connects v_e and v_l.
for each node $v_i \in p_{e,l}$,
for each edge $e_j \in p_{e,l}$ // e_j :{ $(x_j, y_j), (x_k, y_k)$ }
tentatively insert link between v_i and { $median(x_i, x_j, x_k), median(y_i, y_j, y_k)$ }.
for all temporarily inserted link
we finally insert the one with the maximum slack improvement.

Step 5. *Candidate solution pruning*: For a new candidate solution, compare it with previous generated candidate solution for pruning.

If candidate solution $O_{i,k}$ has the exactly same sink set as $O_{i,k}$,
If $C_{i,j} \le C_{i,k}$, $\underline{q}_{i,j} \le \underline{q}_{i,k}$ and $\overline{q}_{i,j} \ge \overline{q}_{i,k}$,
prune $O_{i,k}$

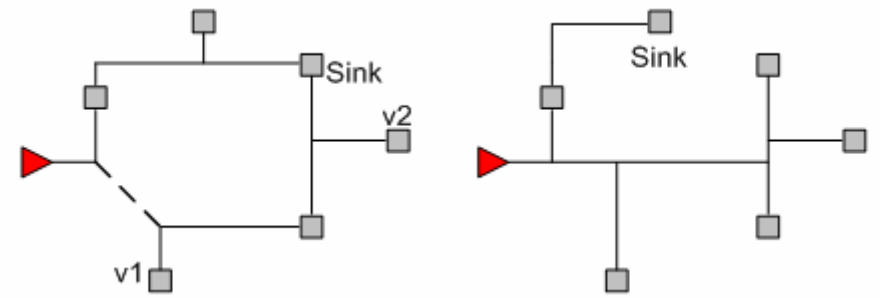
Step 6. *Solutions at the source*: Choose the best solution at the source.

for every solution O_i in the solution set O
choose the one with maximum slack or minimum capacitance without negative slack.

Each step will be explained in detail later.

3.1 Discussion on Topology

The effect of link insertion depends on the initial tree topology. There is area-radius tradeoff among different tree topologies. The area refers to the total wirelength and the radius is the maximum source-sink path length in a tree. The two extreme cases of this trade-off are: (1) chain-like topology (Fig. 2(a)), which has small area and large radius, and is usually derived from minimum spanning tree algorithms; (2) star-like topology (Fig. 2(b)) with relatively large area and small radius, and can be obtained from the shortest path tree or Rectilinear Steiner Aborescence (RSA) algorithms [1].



(a) Chain-like topology (b) Star-like topology

Fig. 2 Chain-like topology and star-like topology.

The major weakness of a tree with chain-like topology is that the delay of some sinks may suffer from the long path length. For example, if v_1 in Fig. 2(a) is the late critical sink with tight delay upper bound, the long detour may cause large delay constraint violation. If we include non-tree topology into consideration, we may reach different conclusions. If a link (dashed line) is inserted in the chain-like topology as in Fig. 2(a), the long detour problem is eliminated and the small wirelength is still enjoyed. However, if the late critical sink is v_2 instead, perhaps the star-like topology in Fig. 2(b) is still better. Thus, it is not clear which tree topology can facilitate a good non-tree solution in general. Our constructive algorithm probes different topologies so that the chance of capturing good non-tree solutions can be increased.

3.2 Constructive Steiner Network Heuristic

If we treat a network as a tree plus links, the problem of network construction can be accordingly decomposed into finding a proper tree topology and link insertions. We combine these two concerns into a dynamic programming based heuristic. This heuristic is a bottom-up merging procedure where multiple candidate solutions are generated to probe good topologies and link insertions. At the beginning, a set of subnetworks are initialized with the sink nodes. In each iteration, a pair of subnetworks is selected to be merged.

Different merging solutions are generated. For each new subnetwork resulting from a merging, another candidate solution is generated by inserting a link in it. These candidate solutions are propagated toward the source.

Solution characterization. A candidate solution O_i is a set of subnetwork $G_{i,j}$. It can be characterized by the total load capacitance $C_{i,j}$, delay lower bound $\underline{q}_{i,j}$ and delay upper bound $\overline{q}_{i,j}$ at each root v_j. It is easy to derive that the delay upper bound $\overline{q}_{i,j}$ is same as the late slack of $G_{i,j}$. Similarly, the delay lower bound $\underline{q}_{i,j}$ is equal to the negative of early slack.

Solution pruning. If there is another candidate solution O_k with the subnetwork set have the exactly same sink sets as candidate solution O_i, the two solutions can be compared for pruning. If for each corresponding subnetwork j has $C_{i,j} \le C_{k,j}$, $\underline{q}_{i,j} \le \underline{q}_{k,j}$ and $\overline{q}_{i,j} \ge \overline{q}_{k,j}$, solution O_k is inferior and can be pruned.

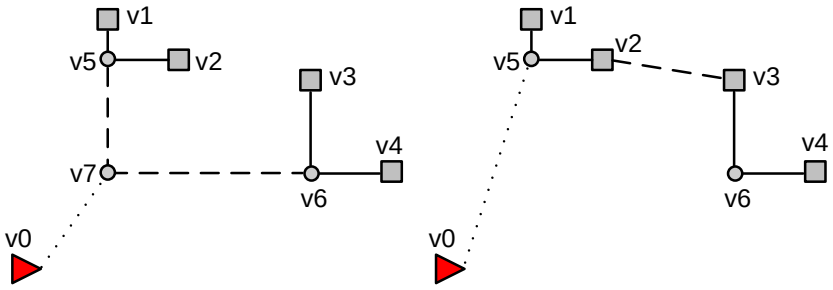
Merging selection. We propose two merging selection criteria for two different scenarios: (1) long path constraints and short path constraints are almost equally tight, and (2) long path constraints dominate.

For the first scenario, we use a merging scheme similar to prescribed skew clock tree routing [9]. In fact, when the delay upper bound of each sink is equal to its delay lower bound, i.e., the delay constraints degenerate to a single value target, this problem is equivalent to prescribed skew clock routing. In prescribed skew clock routing, the subtree with the maximum delay target is merged first to reduce the chance of wire detour [9]. Since we have delay upper and lower bound instead of a single delay target, we use the average $(\overline{q} + \underline{q})/2 + t'$ as the criterion. The t' is the anticipated wire delay from the source

node to the root of the subnetwork. This is to encourage subnetworks with roots far away from the source to be merged early. In each iteration, we first choose the subnetwork with the maximum $(\overline{q} + \underline{q})/2 + t'$, and then merge it with its nearest neighboring subnetwork.

The second scenario is more like traditional signal routing [1]. Therefore, we adopt a merging criterion similar as that of Rectilinear Steiner Aborescence (RSA) [10]. That is, we choose a pair of subnetworks whose merging root is farthest from the source among all pairs. If we consider merging subnetworks rooted at $(x_{i,} y_i)$ and $(x_{j,} y_j)$, then the merging root is at $(x_m = median(x_i, x_j, x_0), y_m = median(y_i, y_j, y_0))$ where (x_0, y_0) is the location of the source node. Then, the pair with the maximum value $|x_m - x_0| + |y_m - y_0|$ is selected for a merging. Our method is different from the well-known RSA algorithm [10] which restricts all sinks in one quadrant if the source is at (0,0). Our merging selection can handle the cases that sinks are distributed in multiple quadrants.

Merging. After a pair of subnetworks is selected, we consider two types of mergings between them. One is the root-root merging as in Fig. 3(a) where subnetwork G_5 and G_6 are merged at node v_7. The other is the shortest merging where two nodes from the two subnetworks with the minimum distance are coneected directly. After the merging, the node closest to the source is selected as the root for the merged network. For example, in Fig. 3(b), the merging between G_5 and G_6 is obtained by connecting v_2 and v_3 where reroot occurs. Then v_5 is chosen as the root.



(a) root-root merging (b) Shortest merging

Fig. 3 Root-root merging and shortest merging.

The root-root merging is very similar as the RSA [10] heuristic which leads to star-like topology. The shortest merging is more likely to result in chain-like topology. By having these two different types of merging, various topologies can be generated to compete for the best slack solution.

Link insertion. For the two subnetworks obtained from merging, we insert a link in each of them such that the slack is maximized considering the double-sided timing constraints.

For the given subnetwork G, which can be either tree or non-tree, we first identify its early critical sink v_e and late critical sink v_l (both defined in Chapter II). Next we find the shortest path $p_{e,l} \in G$ which connects the two critical sinks. For each node $v_i \in p_{e,l}$, we tentatively insert a link between v_i and each edge $e_j \in p_{e,l}$ with the shortest connection. If node v_i is at coordinate (x_i, y_i), and the two ending nodes of e_j are at (x_j, y_j) and (x_k, y_k), respectively, the link is inserted between node v_i and location (x_c, y_c) where $x_c = median(x_i, x_j, x_k)$ and $y_c = median(y_i, y_j, y_k)$. For each link insertion result, we evaluate the slack S of the network. For all temporarily inserted link we finally insert the one that gives the maximum slack improvement.

Solutions at the source. At the source, there are a set of solutions with different capacitance and slack trade-off. We can choose either the maximum slack solution or the minimum capacitance solution without negative slack.

CHAPTER IV

EXPERIMENTAL RESULTS

All algorithms are implemented in C++ and the experiments are performed on a PC computer with 3.2GHz processor and 1G memory. We generated different testcases with the number of sinks ranging from 5 to 25. Without loss of generality, we let the source be at coordinates (0,0). In some cases, all of the sinks are in one quadrant while some other cases have sinks distributed in four quadrants. For example, in the data tables, the notation of "15s, 2Q" means there are 15 sinks and they are distributed in two quadrants. The 70nm technology parameters reported in [12] are employed. We compare the following methods in the experiments:

AHHK. This is a Steiner tree heuristic [1] which can achieve different area-radius tradeoff by varying a parameter $\alpha \in [0,1]$. When the value of α is shifted from 0 to 1, the resulting tree gradually changes from chain-like to star-like topology [1]. Although it is not directly timing driven, we can achieve very good timing performance by trying different α and choosing the result with the best slack. We tested AHHK trees with α = 0, 0.5, 1 in the experiments.

AHHK+detour. If there is short path violation, the edge incident to the early critical sink is elongated to increase the delay till the early slack is close to the late slack, so that the overall slack is maximized.

AHHK+link. Inserting links greedily similar as described in our algorithm step 4 link insertion. The only difference is that here we try to insert links greedily until there isn't timing performing improvement anymore. This method is similar to [3].

Steiner network. The dynamic programming based Steiner network construction

proposed in Chapter IV.

4.1 Cases with Single Critical Sink

For the testcases, we generated 15 nets with 5, 10, 15, 20, 25 sinks and sinks in 1 quadrant, 2 quadrants and 4 quadrants, respectively. Each net has a single critical sink which is often on the long path. Therefore, wire detour is rarely necessary here.

TABLE I
CASES WITH 1 CRITICAL SINK, COMPARISON BETWEEN AHHK AND AHHK+LINK

Case	AHHK			AHHK+link			
	α	S	W	S	W	#L	2-C
15s,1Q	0	-494	8751	-196	10700	1	44%
15s,2Q	0.5	-56	9055	-4	11004	1	42%
15s,4Q	0	-709	15333	-455	17799	1	57%
20s,1Q	0	-704	9887	-248	11963	1	41%
20s,2Q	0	-2137	12453	-1377	14415	1	35%
20s,4Q	0.5	-243	17519	77	19491	1	24%
25s,1Q	0	-746	9596	-442	11290	1	39%
25s,2Q	1	-49	18183	-1	20411	1	33%
25s,4Q	0	-3106	19954	-1749	21612	1	20%
Average		-916	13415	-488	15409	1	37%

Notes: Comparison on slack S(ps), total wirelength W(μm), the number of inserted links #L and percentage of 2-connected wires 2-C.

In Table I above, we compare AHHK and AHHK+link on 9 cases among the 15 nets where links are indeed inserted. The average results in the last row show that link insertion can improve slack by about 428ps with about 15% increase on wirelength. The link insertion can also achieve about 37% 2-connected wires, which means about 37% of the wires are tolerant to open faults.

TABLE II
CASES WITH 1 CRITICAL SINK, COMPARISON BETWEEN AHHK+LINK AND STEINER NETWORK

Case	AHHK+link						Steiner network				
	α	S (ps)	W (μm)	#L	2-C	CPU (s)	S (ps)	W (μm)	#L	2-C	CPU (s)
5s,1Q	0	-3	5122	0	0	0.01	120	7544	1	58%	0.01
5s,2Q	1	-15	7442	0	0	0.01	82	8641	1	30%	0.01
5s,4Q	0	14	9804	0	0	0.01	123	11184	1	25%	0.02
10s,1Q	1	26	7409	0	0	0.01	124	7743	1	15%	0.05
10s,2Q	1	54	12831	0	0	0.01	188	14763	1	42%	0.09
10s,4Q	0.5	112	10120	0	0	0.01	199	13468	3	53%	0.49
15s,1Q	1	102	9566	0	0	0.01	320	8892	0	0	0.25
15s,2Q	1	-13	12135	0	0	0.01	283	13195	1	20%	0.38
15s,4Q	1	-12	18345	0	0	0.01	130	18836	1	22%	0.72
20s,1Q	1	7	12372	0	0	0.02	177	12429	1	19%	2.28
20s,2Q	0.5	-3	14214	0	0	0.02	177	17355	2	39%	4.95
20s,4Q	0.5	77	19491	1	24%	0.02	242	18639	1	19%	0.53
25s,1Q	1	-5	13869	0	0	0.02	534	14493	2	34%	1.41
25s,2Q	1	-1	20411	1	33%	0.02	308	17019	1	18%	10.48
25s,4Q	1	-18	23144	0	0	0.02	211	24128	1	17%	0.66
Average		21	13085		3.8%	0.01	215	13889		27.4%	1.49

Notes: Comparison on slack S(ps), total wirelength W(μm), the number of inserted links #L, percentage of 2-connected wires 2-C and running time CPU(s).

In Table II, we compare our constructive Steiner network heuristic and AHHK+link for the entire 15 nets. For AHHK+link, we pick the results of α with the best timing slack. Among multiple solutions generated by the constructive heuristic, we report the solution with best slack and largest wirelength. According to the last row of Table II, it can improve the slack from 21ps to 215ps on average. The wirelength increase of our Steiner network heuristic is only 6% over the AHHK+link results. The following figures (Fig.4, Fig.5 and Fig.6) show these results.

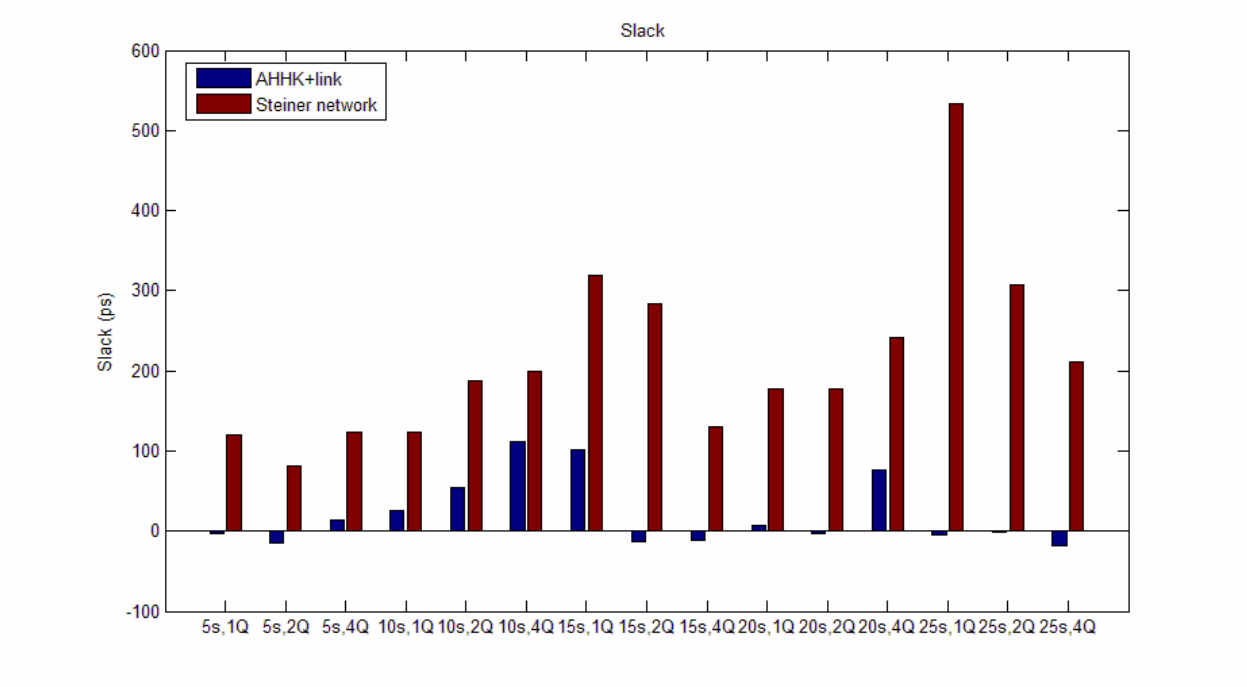


Fig.4 Slack (AHHK+link vs Steiner network).

Notes: With our constructive Steiner network heuristic, we can get better slack result.

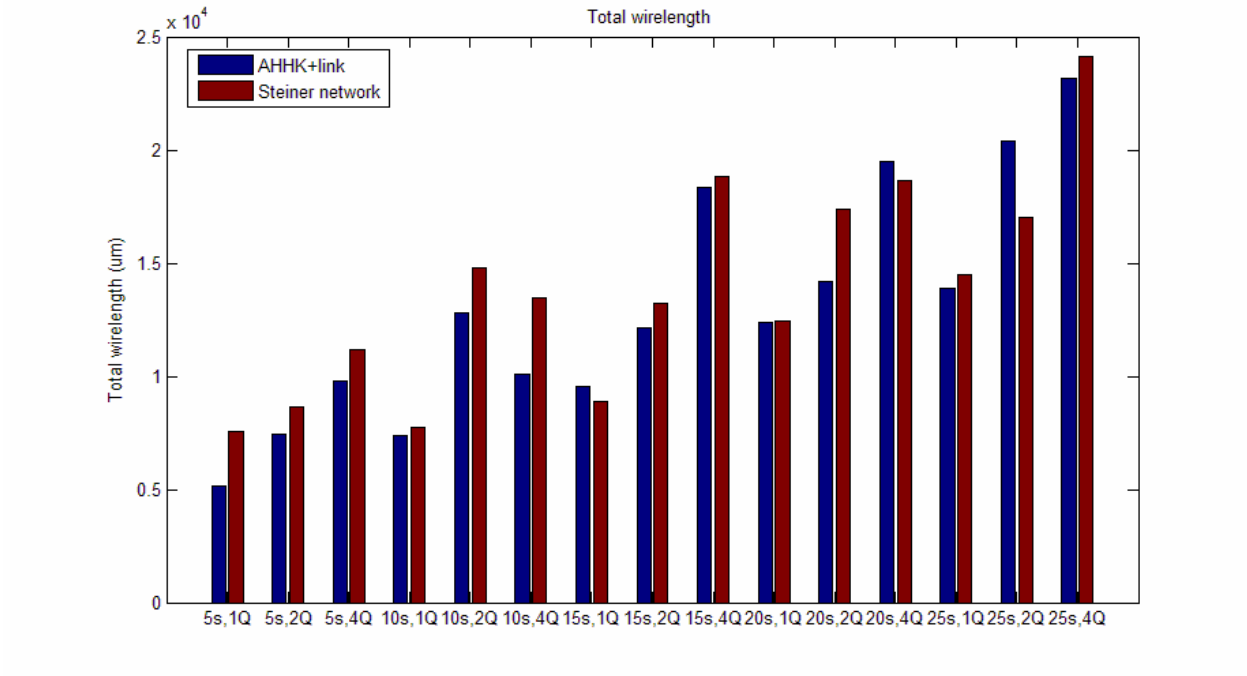


Fig.5 Wirelength (AHHK+link vs Steiner network).

Notes: Comparing with AHHK+link, our Steiner network heuristic has only a little wirelength increase.

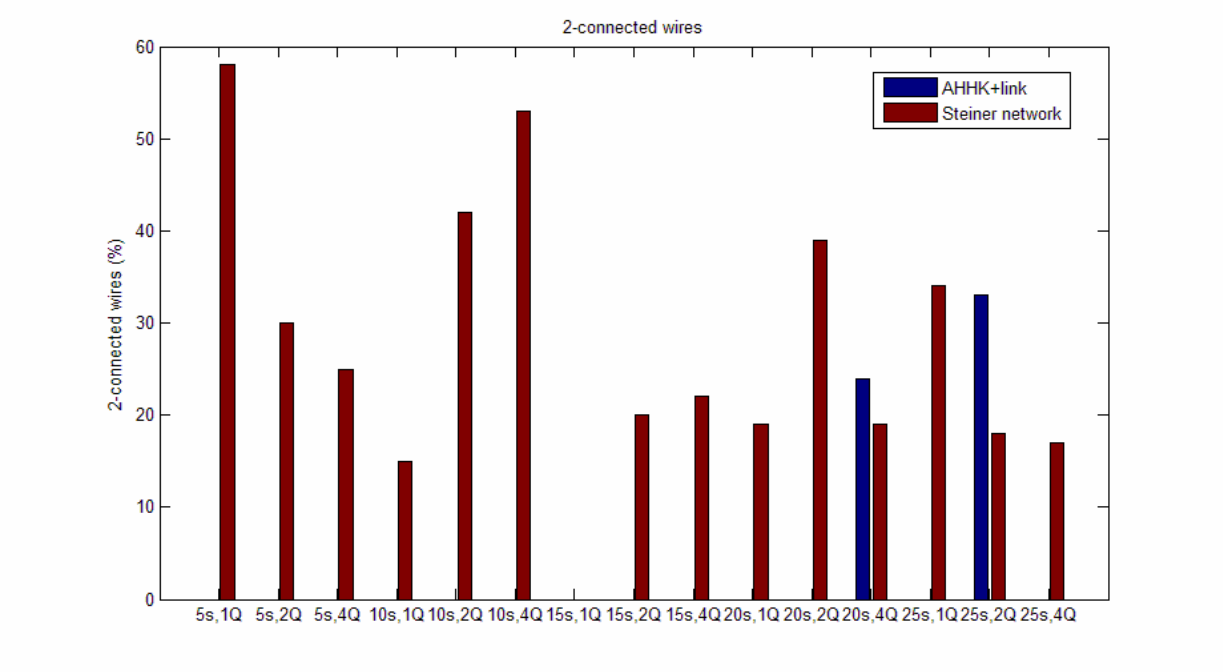


Fig.6 2-connected wire (AHHK+link vs Steiner network).

Notes: In some of our testcases, we didn't insert links in AHHK, therefore there isn't 2-connected wire. While for our constructive Steiner network heuristic, we get more 2-connected wires which can be more tolerant to open faults.

The dynamic programming based Steiner network construction can generate a set of solutions with different slack-wirelength tradeoff.

We also do Monte Carlo simulations (5000 runs for each result) to observe the behaviors of these algorithms under process variations. We consider wire width, sink capacitance and driver resistance variations which are assumed to follow Gaussian distribution with standard deviation equal to 5% of nominal value.

TABLE III
MONTE CARLO RESULTS CORRESPONDING TO TABLE I

Case	AHHK				AHHK+link		
	α	μs	σs	Y	μs	σs	Y
15s,1Q	0	-497	34	0	-197	27	0
15s,2Q	0.5	-57	27	2	-4	27	44
15s,4Q	0	-711	50	0	-456	45	0
20s,1Q	0	-707	35	0	-249	29	0
20s,2Q	0	-2144	69	0	-1382	58	0
20s,4Q	0.5	-245	42	0	77	42	97
25s,1Q	0	-751	40	0	-447	41	0
25s,2Q	1	-51	43	12	-1	44	49
25s,4Q	0	-3117	91	0	-1754	69	0
Average		-920	47.9	1.6%	-490	42.4	21.1%

Notes: mean slack μs (ps), standard deviation of slack σs (ps) and timing yield Y (the probability of non-negative slack).

The comparison between AHHK trees and AHHK+link results is in Table III above. Comparing with the deterministic results in Table I, we can see that the mean values μs of the slacks are about the same. On average, AHHK+link can reduce the standard deviation σs of slack by about 10% and increase timing yield from 1.6% to 21.2%.

As in Table IV, the data indicate that our constructive method can reduce the standard deviation further by about 10% (Fig.7) and improve the timing yield from about 61% to 100% compared to AHHK+link.

TABLE IV
MONTE CARLO RESULTS CORRESPONDING TO TABLE II

Case	AHHK				AHHK+link		
	α	μs	σs	Y	μs	σs	Y
5s,1Q	0	-4	20	43%	119	17	100%
5s,2Q	1	-15	23	25%	82	19	100%
5s,4Q	0	14	27	70%	123	26	100%
10s,1Q	1	24	23	86%	125	22	100%
10s,2Q	1	52	41	89%	189	34	100%
10s,4Q	0.5	111	26	100%	199	29	100%
15s,1Q	1	102	26	100%	318	22	100%
15s,2Q	1	-14	36	35%	283	28	100%
15s,4Q	1	-12	44	39%	130	40	100%
20s,1Q	1	7	29	60%	176	27	100%
20s,2Q	0.5	-5	37	44%	176	39	100%
20s,4Q	0.5	77	42	97%	241	39	100%
25s,1Q	1	-8	38	42%	534	32	100%
25s,2Q	1	-1	44	49%	308	36	100%
25s,4Q	1	-20	54	35%	210	50	100%
Average		21	34.0	60.9%	214	30.7	100%

Notes: mean slack μs (ps), standard deviation of slack σs (ps) and timing yield Y.

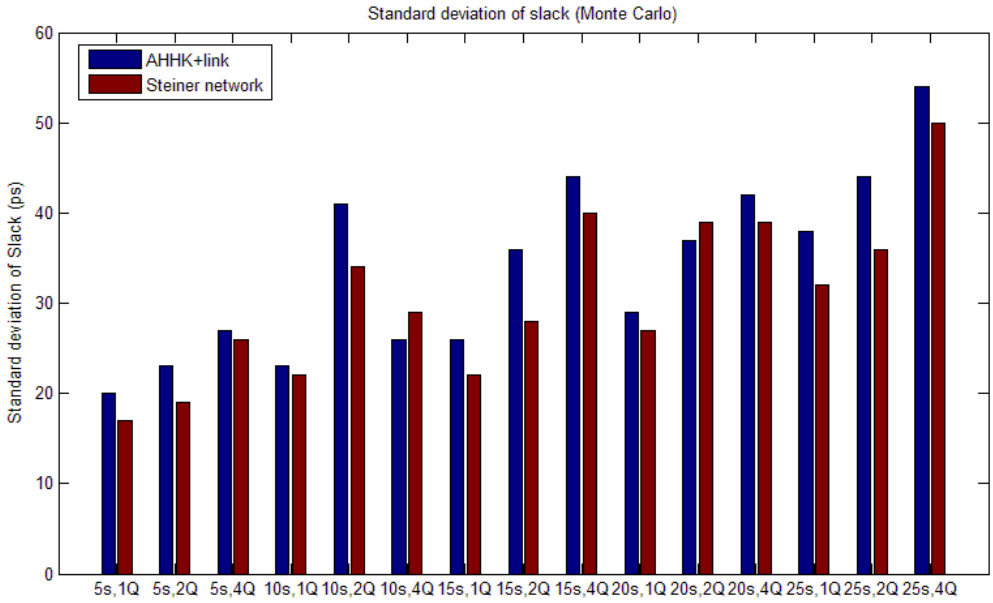


Fig.7 Monte Carlo: standard deviation of slack (AHHK+link vs Steiner network).

Notes: Comparing with the AHHK+link results, our Steiner network heuristic can reduce the standard deviation of slack.

4.2 Cases with Multiple Critical Sinks

We also tested the algorithms in cases with multiple critical sinks. That is, there may be several sinks with similar timing criticality in each net. In order to see the effect on fixing short path delay constraint violations, these testcases usually have tighter constraints on short path than on long path.

The wire detour method can increase the delay to the early critical sink but at the cost of increasing long path delay, while our approach can increase short path delay and reduce long path delay simultaneously. Moreover, wire detour cannot lead to any tolerance to open faults as in non-tree.

The average results in Table V show that our Steiner network heuristic can improve the slack by about 80ps on average when compared to performing wire detour on existing trees. The wirelength increase due to our method is about 4% with respect to the wire detour results.

TABLE V
CASES WITH MULTIPLE CRITICAL SINKS

AHHK		AHHK+detour		AHHK+link		Steiner network	
S	W	S	W	S	W	S	W
-74	14443	218	16677	43	15647	297	17291

Notes: Comparison on average results (10 nets with 5-25 sinks) of slack S(ps), total wirelength W(μm).

CHAPTER V

CONCLUSIONS AND FUTURE WORK

This work investigates timing driven routing by using non-tree topology. We propose a constructive Steiner network heuristic algorithm to do the signal net routing, which can improve the timing performance greatly. Our constructive Steiner network heuristic method considers the double-sided timing constraints, adopts dynamic programming to construct the non-tree solution, and uses greedy link insertion to insert links in subnetwork. And it can handle those cases whose sinks are distributed in multiple quadrants. Experimental results show that this is a very promising approach even when both long path and short path constraints are considered. In future, we can find non-tree routing method using more accurate delay model and study buffered non-tree routings.

REFERENCES

[1] A. B. Kahng and G. Robins. *On Optimal Interconnections for VLSI*. Kluwer Academic Publishers, Boston, MA, 1995.

[2] A. B. Kahng, B. Liu, and I. I. Mandoiu. “Non-tree routing for reliability and yield improvement”. *International Conference on Computer Aided Design*, pp. 260-266, 2002.

[3] B. A. McCoy and G. Robins. “Non-tree routing”. *Design Automation and Test in Europe*, pp. 430-434, 1994.

[4] T. Xue and E. S. Kuh. “Post routing performance optimization via tapered link insertion and wiresizing”. *Design Automation and Test in Europe*, pp. 74-79, 1995.

[5] J. Lillis, C. K. Cheng, T. T. Lin, and C. Y. Ho. “New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing”. *Design Automation Conference*, pp. 395–400, 1996.

[6] W. Chuang, S. S. Sapatnekar, and I. N. Hajj. “Delay and area optimization for discrete gate sizes under double-sided timing constraints”. *Custom Integrated Circuits Conference*, pp. 9.4.1-9.4.4, 1993.

[7] P. K. Chan and K. Karplus. “Computing signal delay in general RC networks by tree/link partitioning”. *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, pp. 898-902, August 1990.

[8] D. Lam, C.-K. Koh, Y. Chen, J. Jain, and V. Balakrishnan. “Statistical based link insertion for robust clock network design”. *International Conference on Computer Aided Design*, pp. 588-591, 2005.

[9] R. Chaturvedi and J. Hu. “An efficient merging scheme for prescribed skew clock routing”. *Transactions on Very Large Scale Integration Systems*, vol. 13, pp. 750-754, June 2005.

[10] S. K. Rao, P. Sadayappan, F. K. Hwang, and P. W. Shor. “The rectilinear Steiner arborescence problem”. *Algorithmica*, vol. 7, pp. 277-288, 1992.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1996.

[12] A. B. Kahng and B. Liu. “Q-Tree: a new iterative improvement approach for buffered interconnect optimization”. in *Proc. IEEE Computer Society Annual Symposium on VLSI*, pp. 183-188, 2003.

VITA

Name:	Qiuyang Li
Address:	Department of Electrical and Computer Engineering
	C/O Dr. Jiang Hu
	Texas A&M University
	College Station, TX 77843-3259
Email Address:	qiuyang@tamu.edu, qiuyang.li@gmail.com

Education: B.S., Computer Science, Nankai University, 1999

M.S., Computer Science, Nankai University, 2002

M.S., Computer Engineering, Texas A&M University, 2006